Transport haute-performance: problématique et solutions proposées

Marc Herbert, Pascale Vicat-Blanc Primet

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Transport haute-performance : problématique et solutions proposées

Marc Herbert, Pascale Vicat-Blanc Primet

N° 4934

septembre 2003

THÈME 1

ISSN 0249-6399 ISRN INRIA/RR--4934--FR

Transport haute-performance : problématique et solutions proposées

Marc Herbert*, Pascale Vicat-Blanc Primet†

Thème 1 — Réseaux et systèmes
Projet RESO

Rapport de recherche n° 4934 — septembre 2003 — 17 pages

Résumé : L'univers du calcul haute-performance a découvert des limitations du protocole TCP, l'empêchant d'utiliser pleinement les liens haut-débit et grande distance prévus pour former l'ossature des *grilles de calcul*. Cet article présente ces problèmes de performance, puis décrit, analyse et compare les solutions actuellement proposées par la communauté réseau.

Mots-clés : Transport, Haute-performance, TCP/IP, Ethernet

* Sun Labs Europe
† INRIA

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38330 Montbonnot-St-Martin (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

High-performance transport: issues and proposed solutions

Abstract: The high-performance computing field has discovered some limitations in the TCP protocol, preventing it to fully use the wide area high capacity links provisioned for computing grids. This article explain the issues, then describes and compares the solutions currently proposed by the networking community.

Key-words: Transport, High-performance, TCP/IP, Ethernet

1 Introduction

Dans la dernière décennie, de nombreux utilisateurs de calcul haute performance ont abandonné les coûteuses architectures spécialisées pour se tourner vers la construction de calculateurs en *grappes*, composées de composants standard dits "sur étagère". Après avoir résolu un grand nombre des problèmes liés à la distribution des calculs sur ce type d'architecture en réseau local, la communauté scientifique se tourne désormais vers le projet ambitieux de virtualiser des ressources de calcul réparties sur des sites géographiquement distants, visant ainsi à constituer ce qu'on appelle une *grille* de calcul. Une difficulté majeure est apparue récemment : les performances décevantes du protocole de transport TCP sur les liens à fort produit débit × délai, et une sous-utilisation des capacités disponibles. Cet article présente le faisceau de causes à la source de ce problème de performance, et dresse un inventaire des solutions actuellement à l'étude. La partie 2 rappelle, parmi les principes des réseaux TCP/IP et Ethernet, ceux qui importent dans un contexte de performance. La partie 3 décrit les problèmes, théoriques et pratiques, rencontrés par TCP sur les réseaux haut-débit. Les parties 4 et 5 décrivent les propositions faites à ce jour par la communauté scientifique pour résoudre ou contourner ces problèmes.

2 Le contexte

Le coût très élevé de conception, fabrication et maintenance des composants matériels et logiciels pousse naturellement vers une grande uniformisation des composants. Dans le domaine des réseaux, cet effet est accentué par les problèmes de compatibilité. Les standards TCP/IP [Ste94] et Ethernet règnent aujourd'hui sans partage sur l'accès aux réseaux de données grande distance : à un point tel que, contrairement aux grappes de calcul, le domaine des grilles n'a même pas pu hésiter entre composants standard et architectures spécialisées.

2.1 Ethernet

Ethernet est désormais le standard incontournable des réseaux locaux, et sa progression s'étend vers les réseaux urbains. Ethernet et IP sont techniquement parfaitement adaptés l'un à l'autre, conçus selon la même sémantique de type datagramme. Avec le récent gigabit/s, Ethernet est également de plus en plus considéré comme une alternative viable dans le domaine du calcul distribué, y compris pour les applications les plus exigeantes en terme de communication.

2.2 TCP/IP

[Cla88] présente les choix de conception fondamentaux de la pile protocolaire TCP/IP. Le choix d'une communication en mode paquet provenait des applications et réseaux pré-existants. Il permettait de plus une meilleure efficacité dans l'utilisation des coûteux liens de l'époque, grâce au multiplexage statistique. Les contraintes de robustesse étant de première priorité, il n'était pas possible de conserver dans les routeurs des données liées aux états des communications en cours (mode *datagramme*), et donc impossible pour le réseau de garantir l'absence de perte ou le séquencement. Ces fonctions devaient donc de fait être assurées par les seules extrémités. Plus généralement, ce choix du mode datagramme, c'est à dire le choix d'un service vraiment *minimal* de la part du réseau, a été fait dans une volonté de flexibilité maximale.

Flexibilité d'une part pour satisfaire des besoins d'interconnexion de réseaux aux caractéristiques très hétérogènes, d'autre part pour être le dénominateur vraiment commun sur lesquels tous les autres services pourraient s'appuyer. De plus, le fait de déplacer les problématiques vers la périphérie du réseau, par nature plus dynamique et évolutive, est un facteur important de passage à l'échelle : les évolutions sont plus simples, même sur grands réseaux. Sans oublier que les coûts de traitement dans le goulet d'étranglement qu'est le réseau (ou tout du moins qu'*était* le réseau) sont minimisés.

Flux d'octets fiable L'unique service réseau offert par le logiciel TCP/IP est celui fourni au travers de la très populaire interface programmatique dite *sockets* BSD. Le service fourni par une socket de type `STREAM` a été conçu initialement pour satisfaire à la fois les besoins de type session de travail à distance et transfert de fichier. Il offre un flux d'octets fiable (ordonné, intègre, sans perte, sans duplicat). À l'exception d'un accès brut aux `DataGRAMmes` (disponible pour réaliser soi-même son transport), on peut donc noter que le service réseau disponible au niveau applicatif est limité à une *unique* sémantique de transport. Dans la pratique ([Coo]), on constate que TCP représente effectivement la majorité du traffic.

Contrôle de congestion L'interface entre le réseau IP et les terminaux se restreint à émission/réception de datagramme, sans aucun autre échange d'information. Se pose alors un problème de gestion des ressources du réseau, celui-ci étant susceptible d'engorgement à cause de la très grande variété de la capacité des liens qui s'y raccordent. La solution choisie [JK88] est une solution coopérative de type *contrôle de congestion* dans laquelle les terminaux émetteurs réduisent par 2 leur débit lorsqu'ils détectent une perte, signe de congestion, et l'augmentent linéairement sinon (AIMD : *Additive Increase Multiplicative Decrease*). [CJ89] démontre la stabilité et l'équité de cet algorithme distribué. La mise en œuvre de cet algorithme est normalisé dans la RFC2581 sous la forme suivante : l'émetteur s'interdit d'injecter dans le réseau plus qu'un certain nombre de paquets non encore acquittés. Cette limite est appelée fenêtre de congestion (*cwnd* dans la suite), et varie de façon linéaire/multiplicative.

3 Problématiques TCP et haut-débit

3.1 Problèmes d'optimisation sur le terminal

Selon [CJRS89] [Chu96] [CGY01] [KRS01], l'essentiel des coûts logiciels TCP/IP ne proviennent pas des traitements protocolaires eux-mêmes mais des mouvements de données (copies[1]) et des changements de contextes, notamment dus aux interruptions provenant de l'interface réseau. Concernant plus précisément les techniques de zéro-copie mémoire, il est intéressant de noter qu'une bonne solution à ce problème avait été trouvée en 1996, mais que celle-ci a été "redécouverte" en 2001, et de façon nettement moins optimale puisque plus aléatoire ("spéculative"). La raison de cette dégradation est la transition d'une technique réseau autorisant des paquets de taille 8 ko (ATM) vers une autre les limitant à 1,5 ko (Ethernet), inférieure à la taille d'une page mémoire. Un autre point notable à ce sujet est la pauvreté fonctionnelle des interfaces matérielles réseau ; une amélioration fréquemment réclamée est qu'elles soient

[1] "A buffer layer can easily grow in complexity to swamp the protocol itself [..] The problem of the buffer layer is made worse by the fact that the protocol specifiers do not admit that such a layer exists" – *in [CJRS89]*

capables de reconnaître et traiter de façon différenciée les paquets reçus. De façon générale il semble que, pour le terminal, le coût logiciel unitaire de communication[2] tend à augmenter avec les progrès techniques [Mar02]. De plus, la majorité des techniques d'optimisation améliorent la performance moyenne mais en la rendant le plus souvent nettement moins prévisible, ce qui peut provoquer des latences voire des pertes rares et inattendues.

3.2 Limitation théorique

Le contrôle de congestion de type AIMD choisi pour TCP introduit fatalement une borne : il faut en effet définir une constante additive pour la phase de croissance linéaire. La constante choisie, normalisée dans la RFC2581, est de 1 segment (de taille SMSS : Sender's Maximum Segment Size) par temps d'aller-retour ou RTT (Round Trip Time). Cette constante détermine une *accélération maximale* du débit en phase de croissance linéaire. Cette agressivité s'exprime simplement à l'aide de quelques hypothèses. Tout d'abord, supposons que le réseau est stable et peu congestionné. En effet, le problème essentiellement constaté dans la pratique est une sous-utilisation de la capacité de réseaux largement dimensionnés. Si le réseau est peu congestionné, le débit est simplement la taille courante de la fenêtre de congestion divisée par le temps qu'il faut pour l'envoyer, ce temps étant quasi constant $tput(t) = cwnd(t)/RTT$. D'après la loi d'accroissement linéaire de la fenêtre de congestion on peut obtenir g, l'accélération maximum du débit :

$$\frac{d\,cwnd}{d\,t}(t) = \frac{SMSS}{b \cdot RTT} \qquad g = \frac{d\,tput}{d\,t}(t) = \frac{SMSS}{b \cdot RTT^2} \tag{1}$$

Le terme b est dû à une particularité de programmation de l'algorithme[3]. Ce facteur pénalisant, toujours supérieur à 1, peut d'un strict point de vue théorique être favorablement pris égal à 1. On peut donc noter que la dynamique du contrôle de congestion de TCP n'est déterminée que par deux paramètres essentiels : latence (au carré) et taille maximum de segment. SMSS est par définition la taille maximum d'un datagramme IP. Il est important à ce stade de préciser qu'en pratique, la taille maximum d'un datagramme IP est égale à la MTU [4] : taille maximum d'un paquet de niveau physique. En effet, la fragmentation IP a été de fait supprimée pour des raisons de performance ([KM87] et RFC1191). Dans le cas d'Ethernet, c'est donc 1.5 ko. Le tableau 1 donne des valeurs typiques[5]. On notera le caractère absolu de ces valeurs fixées par la RFC2581 ; elles dépendent de très peu de paramètres.

La croissance linéaire du débit est simple à exprimer, mais elle n'est pas très parlante ; l'utilisateur est intéressé par le débit moyen sur la durée de sa connexion. La dynamique du débit dépend de la distribution des pertes de paquets, puisque toute perte est supposée signaler une congestion et déclencher la division de *cwnd* — et donc du débit — par deux. Afin d'obtenir un ordre de grandeur du débit en fonction d'un taux de pertes de paquets, [MSMO97] fait l'approximation d'une distribution régulière d'un taux de pertes p de paquets : une perte tous les $1/p$ paquets. Cette dernière approximation n'est pas très réaliste, mais elle permet néanmoins d'obtenir un ordre de grandeur correct et optimiste, comme on peut le vérifier dans [PFTK98]. Nous obtenons alors une courbe de variation de débit régulière en dents de scie, le débit oscillant

[2] c'est à dire le rapport "$\frac{MHz}{Mb/s}$", en simplifiant outrageusement

[3] les "ACK retardés", *cf.* RFC3465

[4] Maximum Transmission Unit

[5] Une fibre optique de 1000 km est parcourue en 5 ms, et on peut grossièrement doubler ce temps pour prendre en compte la latence des équipements réseaux actifs.

RTT (ms)	g ($ko/s/s$)	p
30	1600	$2,3 \times 10^{-7}$
70	306	$4,2 \times 10^{-8}$
100	150	$2,0 \times 10^{-8}$
200	37	$5,1 \times 10^{-9}$

TAB. 1 – accélération, taux de perte max. pour 1Gb/s (SMSS =1.5 ko)

entre $2/3$ et $4/3$ du débit moyen $tput_{mean}$. On peut en déduire le débit moyen en fonction du taux de perte, puis réciproquement, le taux de perte maximum tolérable pour un débit recherché :

$$tput_{mean} \lessapprox \sqrt{\frac{3}{2b}} \frac{SMSS}{RTT\sqrt{p}} \qquad p \lessapprox \frac{3}{2b}\left(\frac{SMSS}{RTT \cdot tput_{mean}}\right)^2 \tag{2}$$

On note que le taux de perte tolérable est en *inverse carré* du débit recherché. L'équation (2), appliqué à un débit recherché de 1 Gb/s, donne les valeurs numériques p du tableau 1.

3.3 Une perte = une congestion ?

Les pertes de paquets peuvent avoir des causes très variées et très difficiles à cerner, la congestion dans le réseau ne représentant que l'une d'entre elles [Par90]. Par exemple les taux de pertes du tableau 1 sont de l'ordre de grandeur des limites de certains composants électroniques et optiques. Elles sont pourtant systématiquement interprétées par TCP comme un signal déclenchant une réduction du débit d'émission. Une perte de paquet de ce type, rare, transitoire, inévitable et sans trace, peut donc par exemple grandement pénaliser les performances. La confusion entre perte et congestion se justifiait parfaitement [JK88] à une époque où la bande passante n'était pas aussi abondante (et donc la congestion plus fréquente). De plus personne n'avait alors anticipé la stagnation du facteur SMSS dans l'équation (2). Notons que les communications TCP utilisant des liens sans-fil doivent faire face à un problème similaire (RFC3155). Une autre source de pertes potentiellement importante, qui semble malgré tout fréquemment négligée, est le terminal lui-même.

3.4 Le terminal est aussi un routeur

Dans le cadre du Principe de "bout en bout" ([SRC84]), [Moo02] remarque que l'extrémité n'est pas le terminal réseau, mais plutôt l'objet logiciel "socket TCP" qui s'y exécute[6]. Il est alors assez naturel de considérer le routeur logiciel IP s'exécutant sur le terminal comme un routeur commutant des paquets de façon similaire à un équipement matériel dédié au routage. La majorité des terminaux TCP/IP est capable d'utiliser simultanément plusieurs connexions réseau, ainsi que d'être connectée à plusieurs réseaux, ce qui nécessite un routeur sur le terminal même. Cette analogie permet aussi de comprendre pourquoi les interfaces réseau n'ont traditionnellement aucune connaissance des connexions : elles font partie du réseau, volontairement réduit au minimum conformément au Principe. Cette analogie est également pertinente lorsqu'on considère les pertes de paquets et les variations de latence. En effet, cette congestion peut se produire dans une file réseau *sur le terminal*, à cause d'une surcharge de travail. Cette surcharge du processeur peut provenir de la charge logicielle du "routeur dans le terminal" ou

[6]Plus précisément : la véritable extrémité réseau est l'application, mais elle délègue ce rôle à TCP.

bien encore, de sa combinaison avec la charge due aux applications, naturellement inconnue. Cette surcharge peut également se produire dans l'interface matérielle d'accès au réseau, dont les limitations en performance sont très variables et inconnues *a priori* [GB02]. De façon générale, toutes les congestions locales au terminal sont interprétées de la même façon que des congestions dans le "vrai" réseau.

On pourrait justifier cette confusion entre pertes locales au terminal et pertes sur le réseau extérieur en argumentant qu'il est légitime de réduire de façon indifférenciée les débits de communication en cas de surcharge du terminal... mais le fait est que cela ne simplifie guère la recherche de performances, puisqu'en mélangeant ainsi indistinctement les problèmes, la recherche des goulets d'étranglements devient alors extrêmement ardue.

4 Modifications sur le terminal

...et sur le terminal uniquement. On trouvera dans cette partie la plupart des solutions proposées par la communauté de recherche aux problèmes de performance exposés ci-dessus. Ces solutions n'impliquent aucun changement dans le réseau, qui continue à s'appuyer sur une technique IP classique. On y trouve notamment des propositions de nouveaux protocoles s'appuyant sur IP. "Utiliser une technique IP classique" ne définit pas pour autant complètement le réseau. En effet, un réseau IP classique s'attend à ce que tous les terminaux coopèrent et régulent leurs flux avec une agressivité égale, conforme à la RFC2581, afin de se partager équitablement le réseau. Les flux TCP classiques conformes à la RFC2581, seront appelés flux de type TCP *Reno* dans la suite. On appellera TCP-*friendly* (aimable avec TCP) un flux ne pénalisant pas plus que n'importe quel autre flux Reno les flux Reno concurrents. On appellera également TCP-*un*friendly et TCP-*nice* les flux perturbant les flux Reno concurrents respectivement plus et moins que s'ils étaient eux-mêmes de type Reno. Déterminer le degré d'amabilité d'un flux est parfois loin d'être évident, particulièrement si sa loi d'évolution de *cwnd* est complexe. Afin d'éviter toute confusion, nous n'utiliserons ici le terme *équité* que lors de comparaisons intra-protocolaires.

4.1 Évolutions de TCP

Un certain nombre de limitations initiales de TCP ont déjà été corrigées dans le passé. Dans la RFC1323, la taille de la fenêtre de réception, anciennement codée avec 16 bits, est désormais codée avec 32 bits et un nouveau champ TCP "datation" permet une mesure précise du RTT, et d'éviter les problèmes de bouclage du numéro de séquence. La RFC2001 formalise les mécanismes *fast retransmit* et *fast recovery* ; ces mécanismes permettent respectivement : – la détection d'une perte sans attendre l'expiration d'une temporisation, – la redémarrage plus rapide de l'émission après une perte. Enfin, la RFC2018 définit un mécanisme d'accusé de réception sélectif (SACK), et non plus uniquement cumulatif. Ces extensions sont disponibles sur tous les systèmes récents, mais nécessitent parfois une intervention de l'administrateur pour être activées.

4.2 Réglages de TCP

Une autre "limitation" de TCP concerne ses réglages par défaut, calibrés dans la majorité des systèmes d'exploitation pour la majorité des utilisateurs, c'est à dire pour des produits

débit × latence largement inférieurs à ceux considérés ici. Le réglage principal concerne la taille des tampons mémoire allouée à chaque connexion TCP. Comme l'API sockets BSD n'offre pas d'émission asynchrone[7], il est en effet nécessaire pour TCP de créer une copie des données qui sont en transit dans le réseau, afin de parer à une éventuelle perte. Ce tampon émetteur doit en fait être largement supérieur au volume de données émises et non encore acquittées, afin d'offrir de plus une certain amortissement à l'interface entre l'application et TCP. De la même manière, le tampon de réception donne cette amortissement à l'autre extrémité. À cause du retard avec lequel l'émetteur en prend connaissance, sa taille sur le récepteur aussi doit prendre en compte le produit débit × délai, et donc au final avoir une valeur comparable à celle de l'émetteur. Ce réglage peut être effectué par un appel système invoquant la fonction "couteau suisse" `setsockopt(fd, SOL_SOCKET. SO_SNDBUF,...)`. Il est fréquent que la taille par défaut de ces tampons soit *inférieure* en pratique à la variable *cwnd*, rendant toute augmentation de cette dernière sans effet. D'autres réglages importants [Dun] concernent par exemple la longueur des files d'attente dans le terminal, leur allongement permettant éventuellement de retarder l'apparition de certaines pertes.

Ce type de réglages pose deux problèmes pratiques importants : 1) comment déterminer la bonne valeur ? En effet, choisir une grande valeur par défaut pour tous les cas conduit à un gaspillage important ; 2) il est nécessaire d'avoir les droits d'administration sur la machine pour les effectuer.

Auto-réglage Le projet web100[8] propose des améliorations du noyau Linux, permettant un accès par l'utilisateur aux variables courantes des connexions TCP, ce qui permet d'analyser en temps réel où se situent d'éventuels goulets d'étranglements. Le projet net100 [Dun] prolonge cette approche en cherchant à adapter de façon automatisée les paramètres des connexions TCP (les paramètres AIMD éventuellement y compris). Indépendamment de ces projets, le noyau Linux contient un certain nombre d'optimisations TCP, en constante évolution, et fréquemment non documentées. Certaines sont à la limite des standards, comme par exemple l'annulation de la réduction de *cwnd* lorsqu'un acquittement arrive "un peu en retard".

4.3 Matériel

Comme le démontre [GB02], les performances des matériels gigabit Ethernet sont extrêmement variables, et de plus très sensibles au pilote logiciel utilisé et à ses *réglages*, auxquels il convient donc d'apporter le plus grand soin. Le gigabit Ethernet, contrairement à son nom, ne définit aucun standard de performance. Lorsque leur performance limite est atteinte, les matériels détruisent silencieusement les trames.

4.4 Connexions parallèles

Un moyen très simple d'augmenter artificiellement l'agressivité de TCP, est tout simplement d'utiliser plusieurs connexions indépendantes, les débits de chacune s'ajoutant. Cette technique présente l'avantage d'être utilisable sans avoir de privilèges d'administration sur le terminal. De plus, elle permet également de multiplier la taille mémoire totale allouée par le système pour cette communication, sans appel système `setsockopt(...)`. Elle s'adapte particulièrement bien aux protocoles autorisant le transfert fractionné des fichiers, comme FTP par exemple ;

[7] c'est à dire en deux temps : 1. émission, *(calcul)* 2. complétion
[8] http://www.web100.org/

dans ce cas, seul une modification du client est nécessaire. Un certain nombre de logiciels l'incluent déjà : `lftp`, `GridFTP`, `bbFTP`,... La bibliothèque `Psockets` permet de factoriser cette modification pour toutes les applications.

Les *sockets parallèles* sont équivalentes du point de vue de l'équation (2) à multiplier SMSS par le nombre de connexions, ce qui est largement TCP-unfriendly : l'agressivité est directement proportionnelle au nombre de connexions ouvertes. D'un autre côté, la RFC2581 n'interdit pas le télé-chargement simultané de plusieurs fichiers depuis un terminal, révélant ainsi une ambiguïté certaine dans les règles de partage de la ressource réseau IP. Cette technique introduit un nouveau paramètre à l'interface entre l'application et TCP : le nombre de connexions parallèles. Les expérimentations de [HA02] montrent que le débit sature au-delà d'un certain nombre à cause d'une croissance des pertes, vraisemblablement due à une congestion croissante. Le choix de ce paramètre reste à ce jour une question en suspens.

4.5 Augmentation de la SMSS

Dans l'équation (2), on note que maintenir le rapport $SMSS/tput_{mean}$, c'est à dire la taille des paquets proportionnelle aux débits recherchés, aurait permis de maintenir naturellement constant le taux de perte tolérable, en faisant croître l'agressivité proportionnellement au débit. Ceci ne s'est pas produit pour deux raisons liées à un souci de compatibilité avec l'existant. Premièrement, faire cohabiter des terminaux à l'agressivité différente est inéquitable. Et enfin, il est impossible de faire cohabiter sur un même réseau Ethernet des équipements avec des MTU différentes (cf. section 5.1).

Une idée intéressante consiste à "tricher" en faisant croire au logiciel TCP qu'il est connecté à un réseau à grande MTU, le re-découpage en segments TCP plus petits étant assuré ensuite par l'interface matérielle. Contrairement aux sockets parallèles ci-dessus, cette technique allège considérablement le coût de communication pour le terminal. Cette technique de *MTU virtuelle* est disponible dans les cartes Intel PRO/1000 récentes. Elle évite toute modification du système d'exploitation.

Il est important de noter à ce stade que la plupart des logiciels TCP travaillent sur une fenêtre de congestion *cwnd* exprimée en segments et pas en octets (*cf.* RFC3465), et font évoluer *cwnd* en fonction du nombre de messages d'acquittement reçus. Si TCP reçoit plusieurs (petits) acquittements pour chaque (gros) segment émis, alors l'évolution de *cwnd* peut ne pas être celle attendue [SCWA99].

4.6 Modifications de l'algorithme AIMD

Les stabilité et équité de ces modifications ont été théoriquement analysées par le projet FAST (section 5.3).

GridDT Un moyen simple d'augmenter l'agressivité de TCP consiste à changer la constante additive $+n\ SMSS/RTT$. Les modifications au noyau Linux proposées par exemple par [Rav03] permettent d'amener cette constante n à une valeur arbitraire au lieu de 1. D'un point de vue de l'agressivité dans l'utilisation du réseau, cette modification est équivalente à ouvrir plusieurs connexions en parallèle.

Highspeed TCP [Flo] vise à augmenter l'agressivité de TCP, mais en restant TCP-friendly en cas de taux de pertes supérieur à 10^{-3}. Les variations de *cwnd* dans Highspeed TCP suivent

donc une loi en deux temps, équivalente à Reno pour les valeurs de *cwnd* inférieures à 38, puis plus agressive au-delà. Au lieu d'un strict mécanisme AIMD, *cwnd* est augmentée d'un coefficient variable ($cwnd = cwnd + a(cwnd)$) en phase de croissance, et divisée en cas de perte ($cwnd = cwnd \times b(cwnd)$) par un coefficient également variable. $a(cwnd)$ et $b(cwnd)$ sont tabulés de telle sorte que l'approximation de l'équation (2) soit remplacée par celle-ci :

$$tput_{mean} \approx 0,12 \frac{SMSS}{RTT \times p^{0,84}}$$

$a(cwnd)$ et $b(cwnd)$ sont calibrés pour atteindre un débit de 10 Gb/s avec un RTT de 100 ms et un taux de perte de 10^{-7}. La stabilité malgré l'abandon d'AIMD est justifiée par le fait que, en cas de congestion, les connexions les plus gourmandes subiront plus de pertes et réduiront donc plus leur *cwnd*. Un prototype est disponible. [THM03] démontre analytiquement et expérimentalement que Highspeed TCP est sujet à de fortes oscillations, et dégrade les performances de flux TCP Reno concurrents lorsqu'il est en phase de haut débit ; "Gentle Highspeed TCP" est alors proposé pour y remédier. Gentle Highspeed TCP surveille les variations de RTT, de façon similaire à Vegas ci-dessous. Il déduit d'une augmentation de RTT un début de congestion et revient alors en mode Reno.

Scalable TCP Scalable TCP [Kel] propose une approche similaire à Highspeed TCP mais plus simple à réaliser. L'algorithme est "Multiplicative Increase" et donc la croissance de *cwnd* est exponentielle dans le temps, indépendamment du débit actuel et de la taille des paquets. La décroissance sur perte est multiplicative. Sur chaque acquittement reçu : $cwnd = cwnd + 0.01$, et sur chaque perte : $cwnd = 0,875 \times cwnd$. Un prototype est disponible. l'approximation de l'équation (2) est remplacée dans les faibles pertes par $tput_{mean} \approx \sqrt{1,5}/(RTT \times \sqrt{p})$.

Vegas TCP Vegas [BP95] a été une étape importante dans le domaine du contrôle de congestion. Pour la première fois a été proposé, simulé et réalisé un algorithme basé sur les variations de RTT afin d'anticiper et prévenir les pertes au lieu de les provoquer. Les lois de variations de *cwnd* visent à conserver constante une certaine quantité de données stockées dans les files d'attente des routeurs. Ces mécanismes permettent d'obtenir une bonne efficacité. En cas de perte, Vegas revient à un comportement de type Reno. Vegas anticipant les pertes, il est légèrement TCP-nice, ce qui peut être vu comme un frein à son déploiement progressif.

TCP-LP [KK03] s'appuie également sur les délais de propagation dans le réseau pour estimer la congestion. Ses spécificités sont : – utiliser les variations de délai *aller* au lieu du RTT, afin de ne pas être bruité par la congestion des liens retour ; – être totalement TCP-nice. Synchroniser les horloges émetteur et récepteur serait bien sûr très délicat, mais ce n'est pas nécessaire puisque seules les *variations* de délai sont utilisées. Cette mesure nécessite par contre une modification sur le récepteur.

4.7 TCP en matériel ?

Une amélioration évoquée depuis longtemps (dès [CJRS89]) et sans vrai succès à ce jour est le fait de déplacer TCP du logiciel vers le matériel, par exemple dans l'interface d'accès au réseau. Un des avantages majeur serait de découpler les problèmes de communication de la charge sur le terminal. Cette optimisation est relativement indépendante de celles présentées

précédemment... si l'on excepte le fait que justement, une réalisation matérielle nécessite une stabilité, une simplicité et une rigueur qui semble assez peu compatible avec la complexité et les fréquentes évolutions décrites ci-dessus.

4.8 Synthèse partielle

Qu'elles réussissent à être TCP-friendly ou non, toutes les propositions s'appuyant sur un réseau IP classique ont un certain nombre de points communs. Toutes continuent à faire reposer la gestion des ressources réseau sur un modèle coopératif sans contrôle d'accès basé sur la confiance. Aucune n'apporte de prévisibilité dans la performance, puisque celle-ci dépend toujours de deux fortes inconnues : la charge présente du réseau mais aussi la charge du terminal. Le fait d'anticiper la congestion grâce aux variations de RTT au lieu d'aller jusqu'aux pertes semble offrir une meilleure efficacité dans l'utilisation du réseau et une plus grande stabilité, au moins théoriquement. On peut par contre s'interroger sur le sur-coût de gestion de ces chronomètres supplémentaires sur le terminal. On peut également être inquiet concernant leur précision. En effet, essayer d'évaluer les variations de latence dans les routeurs du réseau implique que les variations de latence dans le terminal doivent être largement inférieures. Pourtant certaines optimisations locales, comme la coalescence d'interruptions par exemple, sont susceptibles d'y introduire des variations assez importantes et imprévisibles. Il devient alors nécessaire de filtrer ce bruit.

Plus généralement, certaines propositions ajoutent de nouveau(x) paramètre(s) à un logiciel dont la complexité, les fréquentes évolutions et corrections de bogues, laissent déjà de nombreux utilisateurs bien perplexes.

5 Modifications du réseau

Dans cette partie, les propositions impliquant des changements sur le réseau sont présentées. Cela signifie en général que ces propositions ne peuvent *pas* utiliser à ce jour d'équipements réseau du commerce.

5.1 Jumbo frames

Développé initialement par la compagnie Alteon, l'extension de la MTU Ethernet à 9 ko a été normalisée depuis, et connaît un succès relatif. Les jumbo frames présentent des avantages nets en terme de coûts de communication, particulièrement sur le récepteur, et une multiplication mathématique par 6 de l'agressivité de TCP, sans autre modification fondamentale (*c.f.* section 4.5). À noter que le passage de 1,5 ko à 9 ko simplifie considérablement les optimisations décrites en section 3.1 grâce au franchissement d'un seuil important : la taille d'une page mémoire. On note que la multiplication par 6 en 20 ans de la taille de la MTU Ethernet, même si elle va dans le bon sens, est assez ridicule face à la multiplication par 100 et bientôt 1000 du débit nominal dans le même laps de temps. Malheureusement, Ethernet est utilisé pour créer des réseaux de plus en plus grands, reliant sans routage explicite des nombres croissants de terminaux ; ceci rend alors plus difficile la bascule d'un réseau Ethernet vers l'utilisation de trames plus grandes.

5.2 ECN

ECN (RFC3168, standards track) est une modification globale mais assez légère, qui est déjà relativement populaire dans les réseaux IP en service. Il s'agit pour les routeurs, lorsque les terminaux et autres routeurs sont compatibles, de signaler à l'émetteur la congestion d'un lien traversé, en utilisant le récepteur comme relais. L'émetteur traite alors ce signal de la même façon qu'une perte, mais avec le coût de retransmission en moins.

5.3 FAST

Le projet FAST [FAS] a effectué une analyse approfondie de tous les mécanismes de contrôle de congestion présentés ici (Reno, Vegas [LPW02]) et de leurs variantes en utilisant les outils de la théorie de contrôle des systèmes dynamiques. Leurs limitations respectives ont été démontrées analytiquement, et des corrections proposées. Leur principale proposition originale [PWLD03] est basée d'une part sur une information de coût par lien remontant continûment du réseau vers l'émetteur, de façon similaire à ECN mais reposant sur un calcul un peu plus élaboré au niveau des liens[9] ; et d'autre part sur une régulation coopérative de la part des émetteurs qui cherchent à minimiser ces coûts. Cette approche permet d'obtenir de très bons résultats théoriques en terme de performance (efficacité, taux de perte,...). Divers prototypes, adaptés à diverses richesses d'information remontant depuis le réseau, sont en cours de développement et de test.

5.4 XCP

XCP [KHR02], de la même façon qu'ECN, utilise le récepteur comme un relais pour que les liens congestionnés puissent transmettre à l'émetteur une information de congestion. Contrairement à ECN, l'information calculée dans les routeurs et remontée à l'émetteur n'est pas codée de façon seulement binaire. Chaque connexion marque tous ses paquets avec sa *cwnd* et son estimation de RTT courants, ces valeurs étant ajustées par le routeur conformément à sa congestion courante. Lorsqu'un paquet traverse plusieurs routeurs, c'est l'ajustement le plus limitant (goulet d'étranglement) qui est retenu. Une innovation importante est le découplage des problèmes d'équité et d'efficacité d'utilisation des liens, grâce à un calcul en deux temps. C'est ce qui permet d'offrir aux flux une forte agressivité dans le remplissage des liens, tout en conservant un mécanisme d'équité basé sur l'AIMD. Plusieurs techniques de réseaux hybrides XCP/TCP sont suggérées dans un but de déploiement progressif. XCP est au stade de la validation par simulation.

5.5 Commutation de circuits

Veeraraghavan et al. proposent dans [VLZ03] et [VKK+01] d'ouvrir dynamiquement des liaisons optiques unidirectionnelles point à point Ethernet sur SONET, afin de transférer des fichiers volumineux à débit constant, se débarrassant ainsi du problème de congestion. La signalisation, de type G-MPLS, peut être initiée par un réseau IP classique, qui peut être également utilisé pour retransmettre les rares paquets erronés. Le protocole de transport considéré est ANSI ST (Scheduled Transfer Protocol, [Pie01]), issu de l'univers réseau local HiPPI/GSN. Le débit moyen

[9] REM : *Random Exponential Marking*

	signal de congest.	TCP friendly	réglage complexe	coût CPU	stable / prévis.	disponible	Tb/s	larges réseaux
Auto-réglage	*n/a*	*n/a*	+	-	- -	+	-	0
// sockets		- -	-	- -	- -	+ +	- -	0
Virtual MTU		- -	0	+ +	-	0	-	0
GridDT	perte	- -	-	+	0	+	-	0
Highsp. TCP	perte	?	-	-	-	+	-	0
Scalable TCP	perte	?	0	0	-	+	-	0
Vegas	RTT	+	0	-	+	+	0	0
TCP-LP	latence	+ +	0	-	-	+	0	0
Jumbo f.	*n/a*	-	0	+	0	-	0	-
ECN	ECN	*n/a*	0	+	+	0	+	0
FAST/REM	REM	+	0	+	+	-	+	0
XCP	XCP	*n/a*	0	+	+	- -	+	0
Circuits	*n/a*	*n/a*	-	+	+ +	- -	++	- -

du transfert dépend du temps de latence de l'ouverture du circuit optique, qui peut être important ; cette technique se destine clairement en priorité aux transferts très volumineux. Afin de réduire le volume minimal rentable pour l'ouverture d'un circuit, Veeraraghavan et al. prototypent sur FPGA une accélération matérielle de la signalisation. Le projet TCPSwitching étudie également la viabilité de la commutation de circuits pour transporter des données. La principale approche retenue est l'ouverture implicite de circuits à l'arrivée de paquets au niveau du cœur de réseau ; [MFM03] démontre que le multiplexage statistique n'apporte pas *systématiquement* une meilleure efficacité globale. Le réseau d'éducation et recherche canadien CANARIE cherche à offrir à ses utilisateurs la possibilité d'ouvrir eux-mêmes des circuits optiques de la façon la plus dynamique possible [Arn].

5.6 Bilan

Deux familles d'évolutions du réseau se distinguent nettement. FAST et XCP plaident pour déplacer la réalisation du contrôle de congestion légèrement du terminal vers le réseau, en enrichissant un peu les échanges d'informations entre eux. Le réseau conserve son caractère fondamentalement sans état. Ce déplacement doit malgré tout permettre de simplifier le logiciel de transport sur le terminal, d'y diminuer le coût de communication, et de rendre la performance plus stable et plus prévisible. La commutation de circuits prône au contraire une évolution très marquée, et fait le pari qu'il peut être rentable pour certains transferts et certains réseaux d'avoir un réseau avec états, grâce aux récents progrès réalisés par les techniques de gestion des réseaux opérateurs. Le réseau gère explicitement des connexions au profit du protocole de transport, qui peut alors être débarrassé de la majorité de sa complexité. Dans le contexte des grilles de calcul, il est possible que ces approches puissent toutes deux trouver un terrain d'application, en fonction de la taille du réseau et du type de services recherchés. Le tableau ci-contre évalue les différentes solutions recensées ici (au risque de comparer "des pommes et des oranges"). De nombreuses solutions ne sont *pas* mutuellement exclusives. La référence 0 correspond à la situation actuelle. Un "+" est toujours signe de *mieux* ; un "+" en "coût CPU" ne veut *pas* dire "+ de coût CPU".

6 Conclusion

Les choix de conception de TCP/IP [Cla88] plaçaient la performance en très faible priorité. La recherche de performance en souffre aujourd'hui de plusieurs façons : les taux de perte de paquets ont une variance très élevée, et leurs causes sont très diverses. Le fait que le réseau fournisse un

service et une signalisation minimaux aux terminaux rend difficile les optimisations. Un grand nombre de propositions réclame un échange d'information entre le réseau et les terminaux plus riche. La complexité, entièrement supportée par les terminaux, rend leur comportement délicat à stabiliser et à prédire, alors qu'ils doivent déjà faire face aux aléas du réseau. Certaines grilles de calcul ont parfois la chance de disposer de réseaux expérimentaux et de taille maîtrisée, ce qui leur offre une latitude leur permettant de mettre en œuvre des solutions pouvant plus facilement s'affranchir de la compatibilité avec l'existant. Une technique simple déjà très populaire dans les grappes pourra peut-être aider les grilles à résoudre leurs problèmes réseau : la double connectivité, constituée 1) d'un réseau classique bas débit de contrôle et de signalisation 2) d'un réseau dédié à la performance et à l'expérimentation. On peut par exemple constater que cette double connectivité simplifie considérablement la mise en œuvre de solutions très simples et efficaces comme l'augmentation de MTU ou le changement de constantes AIMD, puisqu'on s'affranchit ainsi de la contrainte d'amabilité avec TCP.

Remerciements

Merci à Benjamin Gaidioz pour les innombrables et enrichissantes discussions à propos des algorithmes de contrôle de congestion. Ce travail a été financé par le ministère français de la recherche (ACI GRID) et Sun Microsystems.

Table des matières

1 Introduction **3**

2 Le contexte **3**
2.1 Ethernet 3
2.2 TCP/IP 3

3 Problématiques TCP et haut-débit **4**
3.1 Problèmes d'optimisation sur le terminal 4
3.2 Limitation théorique 5
3.3 Une perte = une congestion ? 6
3.4 Le terminal est aussi un routeur 6

4 Modifications sur le terminal **7**
4.1 Évolutions de TCP 7
4.2 Réglages de TCP 7
4.3 Matériel 8
4.4 Connexions parallèles 8
4.5 Augmentation de la SMSS 9
4.6 Modifications de l'algorithme AIMD 9
4.7 TCP en matériel ? 10
4.8 Synthèse partielle 11

5 Modifications du réseau **11**
5.1 Jumbo frames 11
5.2 ECN 12
5.3 FAST 12
5.4 XCP 12
5.5 Commutation de circuits 12
5.6 Bilan 13

6 Conclusion **13**

Références

[Arn] Bill St. Arnaud. Customer-owned networks. in LightReading. http://www.lightreading.com/document.asp?doc_id=20448.

[BP95] L. Brakmo and L. Peterson. TCP vegas : End to end congestion avoidance on a global internet. *IEEE Journal on Selected Areas in Communication*, 13 :1465–1480, November 1995.

[CGY01] Jeffrey S. Chase, Andrew J. Gallatin, and Kenneth G. Yocum. End system optimizations for high-speed TCP. *IEEE Communications Magazine*, 39 :68–74, May 2001.

[Chu96] H.K. Jerry Chu. Zero-copy TCP in solaris. In *Proc. of USENIX*, February 1996.

[CJ89] Dah-Ming Chiu and Raj Jain. Analysis of the increase/decrease algorithms for congestion avoidance in computer networks. *Computer Networks and ISDN Systems*, 17(1) :1–14, June 1989.

[CJRS89] David D. Clark, Van Jacobson, John Romkey, and Howard Salwen. An analysis of TCP processing overhead. *IEEE Communications Magazine*, 27 :23–29, July 1989.

[Cla88] David D. Clark. The design philosophy of the DARPA internet protocols. *SIGCOMM Computer Communication Review*, 18 :106–114, August 1988.

[Coo] Cooperative Association for Internet Data Analysis. Characterization of internet traffic loads, segregated by application. http://www.caida.org/analysis/workload/byapplication/.

[Dun] Tom Dunigan. Dunigan's network performance links. http://www.csm.ornl.gov/~dunigan/netperf/.

[FAS] FAST project web site. http://netlab.caltech.edu/FAST/.

[Flo] Sally Floyd. Highspeed TCP web page. http://www.icir.org/floyd/hstcp.html.

[GB02] Paul Gray and Anthony Betz. Performance evaluation of copper-based gigabit ethernet interfaces. In *Proc. of 27th Annual IEEE Conference on Local Computer Networks (LCN'02)*, Tampa, Florida, November 2002.

[HA02] Thomas J. Hacker and Brian D. Athey. The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network. In *Proceedings of the 16th International Parallel and Distributed Processing Symposium*, May 2002.

[JK88] Van Jacobson and Michael Karels. Congestion avoidance and control. In *Proceedings of SIGCOMM'88*, Stanford, CA, August 1988.

[Kel] Tom Kelly. Scalable TCP. http://www-lce.eng.cam.ac.uk/~ctk21/scalable/.

[KHR02] Dina Katabi, Mark Handley, and Chalrie Rohrs. Congestion control for high bandwidth-delay product networks. In *Proc. of SIGCOMM*, pages 89–102. ACM, September 2002.

[KK03] Aleksandar Kuzmanovic and Edward W. Knightly. TCP-LP : A distributed algorithm for low priority data transfer. In *Proc. of IEEE INFOCOM*, San Francisco, April 2003.

[KM87] C. Kent and J. Mogul. Fragmentation considered harmful. *SIGCOMM Computer Communication Review*, 17(5), August 1987.

[KRS01] Christian Kurmann, Felix Rauch, and Thomas M. Stricker. Speculative defragmentation - leading gigabit ethernet to true zero-copy communication. *Cluster Computing Journal : The Journal of Networks, Software Tools and Applications.*, 4 :7–18, April 2001.

[LPW02] Steven H. Low, Larry Peterson, and Limin Wang. Understanding vegas. *Journal of ACM*, 49(2) :207–235, March 2002.

[Mar02] Evangelos Markatos. Speeding up TCP/IP : Faster processors are not enough. In *Proc. of International Performance, Computing and Communications Conference*, Phoenix, Arizona, April 2002. IEEE.

[MFM03] Pablo Molinero-Fernández and Nick McKeown. The performance of circuit switching in the internet. *OSA Journal of Optical Networking*, 2(4) :83–96, March 2003.

[Moo02] Tim Moors. A critical review of end-to-end arguments in system design. In *Proc. of International Conference on Communications (ICC2002)*, April 2002.

[MSMO97] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communication Review*, 27(3), July 1997.

[Par90] Craig Partridge. How slow is one gigabit per second ? *SIGCOMM Computer Communication Review*, 20, February 1990.

[PFTK98] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. Modeling TCP throughput : a simple model and its empirical validation. In *Proc. of ACM SIGCOMM*, September 1998.

[Pie01] Pekka Pietikäinen. Hardware-assisted networking using scheduled transfer protocol on linux. Master's thesis, University of Oulu, Finland, December 2001. http://www.netppl.fi/~pp/.

[PWLD03] Fernando Paganini, Zhikui Wang, Steven Low, and John Doyle. A new tcp/aqm for stable operation in fast networks. In *Proceedings of INFOCOM*, San Francisco, April 2003. IEEE.

[Rav03] Sylvain Ravot. GridDT. In *Proceedings of PFLDnet*. CERN, February 2003.

[SCWA99] Steven Savage, Neal Cardwell, David Wetherall, and Tom Anderson. TCP congestion control with a misbehaving receiver. *SIGCOMM Computer Communication Review*, 29 :71–78, November 1999.

[SRC84] Jerome H. Saltzer, David] P. Reed, and David. D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4) :277–288, November 1984.

[Ste94] W. Richard Stevens. *TCP/IP Illustrated, volume 1.* Addison-Wesley, 1994.

[THM03] Koichi Tokuda, Go Hasegawa, and Masayuki Murata. Performance analysis of highspeed tcp and its improvement for high throughput and fairness against tcp reno connections. In *Proc. of High-Speed Networking Workshop*, San Francisco, March 2003. IEEE.

[VKK+01] Malathi Veeraraghavan, Mark Karol, Ramesh Karri, Tim Moors, and Reinette Grobler. Architectures and protocols that enable new applications on optical networks. *IEEE Communications Magazine*, 39(3) :118–127, March 2001.

[VLZ03] Malathi Veeraraghavan, H. Lee, and X. Zheng. File transfers across optical circuit-switched networks. In *Proceedings of PFLDnet*. CERN, February 2003. http://datatag.web.cern.ch/datatag/pfldnet2003/program.html.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38330 Montbonnot-St-Martin (France)

Unité de recherche INRIA Futurs : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399